

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-067817

(43)Date of publication of application : 11.03.1994

(51)Int.Cl.

G06F 3/08

G06F 3/06

(21)Application number : 04-223827

(71)Applicant : FUJITSU LTD

(22)Date of filing : 24.08.1992

(72)Inventor : KURIHARA TAKUYA

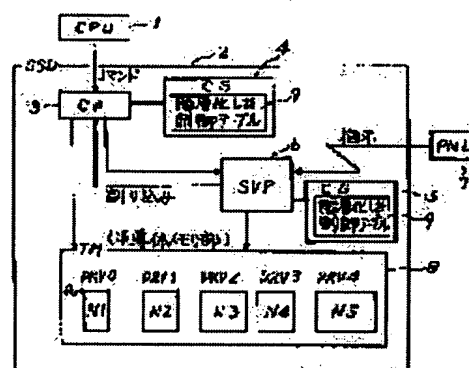
(54) METHOD FOR MAINTAINING SEMICONDUCTOR DISK DEVICE

(57)Abstract:

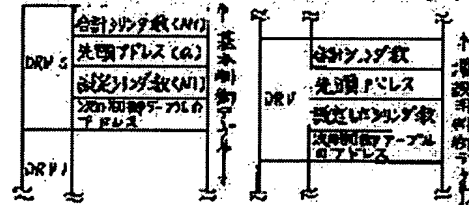
PURPOSE: To improve the operation efficiency of a device by executing processing without turning off the device at the time of changing memory capacity or repairing a fault memory.

CONSTITUTION: In the maintenance method of a semiconductor disk device 2 to be used by dividing a semiconductor memory part (TM) 8 into plural logical drives DRV0 to DRV1, hierarchical control tables 9 each of which consists of a basic control table and an extended control table are set up in the device 2 at every logical drive, and at the time of changing a memory, the control table 9 is updated at every logical drive, so that memory changing processing can be executed so as to make access logical drives other than the one concerned from a master device 1.

A: 半導体ディスク装置のブロック図



B: 制御テーブル (階層化制御テーブル) の構成



LEGAL STATUS

[Date of request for examination] 25.06.1996

[Date of sending the examiner's decision of rejection] 26.05.1998

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection] 10-09954

[Date of requesting appeal against examiner's decision of rejection] 25.06.1998

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

【特許請求の範囲】

【請求項1】 記憶媒体として半導体メモリ（TM8）を使用し、

この半導体メモリ（TM8）を、所定の容量を持つ複数の論理ドライブ（DRV0、DRV1、・・・）に分割して使用することにより、

ディスク装置の入出力処理を仮想的に実行する半導体ディスク装置（2）の保守方法において、

該半導体ディスク装置（2）の内部に、上記論理ドライブの制御情報格納用の制御テーブル（9）を、各論理ドライブ（DRV0、DRV1、・・・）毎に設定すると共に、

この制御テーブル（9）を、階層化したテーブルとして設定し、

メモリの変更時に、各論理ドライブ（DRV0、DRV1、・・・）毎に、上記制御テーブル（9）を更新処理することにより、

該当する論理ドライブ以外の論理ドライブを、上位装置（1）からアクセス可能な状態で、メモリ変更処理を行うことを特徴とした半導体ディスク装置の保守方法。

【請求項2】 上記制御テーブル（9）を、制御情報の初期設定値を格納する基本制御テーブルと、該基本制御テーブルと同じ構成の増設用制御テーブルとで構成し、

これら基本制御テーブルと、増設用制御テーブルに、次の制御テーブルのアドレスを設定することにより、階層化したことを特徴とする請求項1記載の半導体ディスク装置の保守方法。

【請求項3】 オンラインで稼働中の上記半導体ディスク装置において、メモリを増設する際、

半導体ディスク装置（2）のプロセッサ（3、6）では、外部からの指示により、上記制御テーブル（9）を参照して、増設する論理ドライブの制御情報を獲得し、増設する論理ドライブのみを、オフラインにして、増設部のフォーマット処理を行うと共に、

増設した論理ドライブの制御テーブル（9）に、制御情報の値を再定義し、

その後、増設部の論理ドライブをオンラインに戻すことを特徴とした請求項1記載の半導体ディスク装置の保守方法。

【請求項4】 オンラインで稼働中の上記半導体ディスク装置において、障害の発生したメモリの修復（交換）処理を行う際、

半導体ディスク装置（2）のプロセッサ（3、6）では、外部からの指示により、上記制御テーブル（9）を参照して、修復する論理ドライブの制御情報を獲得し、修復する論理ドライブのみをオフラインにして、外部へ通知し、

その後、障害メモリの交換終了の指示を受けたプロセッサ（3、6）では、

上記制御テーブル（9）を参照して、オフラインにした論理ドライブのフォーマット処理を行い、

該フォーマット処理終了後、該当する論理ドライブを、オンラインに戻すことを特徴とした請求項1記載の半導体ディスク装置の保守方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、記憶媒体として半導体メモリを使用し、磁気ディスク装置の入出力処理を、仮想的に実行する半導体ディスク装置の保守方法に関する。

【0002】

【従来の技術】図12～図14は、従来例の説明図であり、図12Aは半導体ディスク装置のブロック図、図12Bは制御テーブル例である。また、図13はシリンダ数を増やす場合の説明図であり、図13Aはフォーマット前、図13Bはフォーマット後を示す。更に、図14はメモリ障害の修復時の説明図である。

【0003】図12～図14中、1は中央処理装置（Central Processing Unit：以下「CPU」と記す）、2は半導体ディスク装置（Solid State Disk：以下「SSD」と記す）、3はチャネルアダプタ（Channel Adapter：以下「CA」と記す）、4、5はコントロールストレージ（Control Storage：以下「CS」と記す）、6はサービスプロセッサ（以下「SVP」と記す）、7はメンテナンスパネル（以下「PNL」と記す）、8トラックメモリ（Track Memory：以下「TM」と記す）、9は制御テーブルを示す。

【0004】従来、記憶媒体に半導体メモリを使用し、ディスク装置の入出力処理を仮想的に実行する装置として、SSDが知られていた。その1例（例えば、特開昭61-165153号公報参照）を図12に示す。

【0005】①：SSDの構成の説明・・・図12参照図示のように、SSD2はCA3、SVP6、CS4、CS5、TM8、PNL7等で構成され、ホストシステムであるCPU1に接続して使用される。

【0006】このSSD2では、所定の容量を持つTM8（半導体メモリ部を構成）を、複数の容量（例えば、N1～N5）に分割し、それぞれ独立した論理ドライブ（DRV0～DRV4）と見なすこと（エミュレート機能）により、データの入出力処理を行っている。

【0007】上記CA3はCPU1との間で、コマンド或いはデータのやりとり（入出力処理）や、SSD2のエミュレート機能をサポートするプロセッサであり、SVP6はSSD2の全体を制御するプロセッサである。

【0008】PNL7はオペレータ等により操作されるパネルであり、外部より、SVP6に対して指示を出し、保守を行わせるものである。CS4及びCS5内に

は、それぞれ、図12Bに示したような制御テーブル9が設定されており、CS4内の制御テーブル9をCA3が使用し、CS5内の制御テーブル9をSVP6が使用することにより、各種の制御（エミュレート）を行う。

【0009】TM8は半導体メモリ部を構成するメモリであり、例えば、図示のように、容量の異なる複数の論理ドライブDRV0～DRV4に分割して制御を行う。この場合、上記のように、CS4及びCS5内に、制御テーブル9を設定し、各論理ドライブDRV0～DRV4毎の容量（シリンダ数：CLYNUM）を登録しておく。

【0010】この例では、論理ドライブDRV0～DRV4に対し、その容量を、DRV0=N1、DRV1=N2、DRV2=N3、DRV3=N4、DRV4=N5のように設定する。

【0011】②：論理ドライブの容量を変更（この例では増設）する場合の処理説明・・・図12、図13参照
上記のようにTM8における論理ドライブDRV0～DRV4の容量（シリンダ数）を、それぞれ、N1～N5に設定した状態で使用していたとする。この状態で、容量（シリンダ数）を増加する場合、次のようにする。

【0012】②-1：増設するメモリを実装する。

②-2：CPU1より、特定の命令をSSD2に対して発行する。この特定の命令により、全ての論理ドライブのシリンダ数を指定する。

【0013】②-3：上記命令により、CA3は、SVP6に対して、制御テーブル9の設定変更を指示する。

②-4：SVP6は、一旦配下の記憶領域であるTM8内の各論理ドライブDRV0～DRV4をオフラインにし、CS5内の制御テーブル9を、上記命令によって再定義する。

【0014】②-5：上記処理が終了した時点で、SVP6は、割り込みにより、CA3に、再定義処理の終了を通知する。

②-6：CA3は、上記割り込みを受け付けた後、CS4内の制御テーブル9を、上記命令に従って再定義し、制御テーブル9に基づいて、TM8の再フォーマットを行う。

【0015】なお、上記のように、CPU1より特定の命令を発行する代わりに、PNL7より変更指示を出し、上記②-4～②-6の処理を行わせる方法もある。以上の処理により、メモリの容量変更を行う。この場合、図13に示したように、フォーマット前は、各論理ドライブDRV0～DRV4の各容量（シリンダ数）は、DRV0=N1、DRV1=N1、DRV2=N2、DRV3=N4、DRV4=N5であり、これに、DRV0=N9、DRV1=N10、DRV2=N11の各容量を増設する。

【0016】上記処理が終了したフォーマット後は、各論理ドライブDRV0～DRV4の各容量（シリンダ

数）は、DRV0=N1+N9、DRV1=N1+N10、DRV2=N2+N11、DRV3=N4、DRV4=N5となる。

【0017】③：メモリ障害時の修復時の処理説明・・・図12、図14参照

TM8に障害が発生した場合、次のようにして処理を行う。

③-1：例えば、PNL7からの指示により、SVP6は、配下の論理ドライブDRV0～DRV4をオフラインにする。

【0018】③-2：障害の発生したメモリ部（例えば、DRV2）を、良品と交換する。

③-3：SVP6は、制御テーブル9をCS5内に設定すると共に、この処理が終了した時点で、SVP6は、割り込みにより、CA3に、再定義処理の終了を通知する。

【0019】③-4：CA3は、上記割り込みを受け付けた後、CS4内の制御テーブル9を再定義し、制御テーブル9に基づいて、TM8の再フォーマットを行う。

【0020】

【発明が解決しようとする課題】上記のような従来ものにおいては、次のような課題があった。

(1)、上記論理ドライブの容量を変更（シリンダ数の増設等）する場合、CPU1又は、PNL7からの設定情報を使用し、全ての論理ドライブのシリンダ数を設定／更新する。

【0021】そのため、全てのドライブ（装置全体）を一度オフライン状態とし、増設するメモリと共に、それまで使用していたメモリを、CA3により、全て再フォーマット処理する。

【0022】すなわち、1つの論理ドライブに付き、1つの制御テーブルで制御するような方法では、容量の変更を行う際、上記のように、SSD2を全てCPU1から切り離した後、全てのメモリを再フォーマット化しなければならない。

【0023】従って、SSD2が連続して使用出来なくなるため、装置（SSD）の稼働効率（利用効率）が低下する。

(2)、メモリ障害が発生した場合においても、上記容量変更時と同様に、SSD2内の全ての論理ドライブを、一度オフライン状態にして、修復処理を行わなければならない。

【0024】従って、この場合にも、装置（SSD）の稼働効率（利用効率）が低下する。本発明は、このような従来課題を解決し、メモリ容量の変更、或いは、障害メモリの修復時に、装置をオフラインにすることなく、処理が出来るようにして、装置の稼働効率（利用効率）を向上させることを目的とする。

【0025】

【課題を解決するための手段】図1は本発明の原理説明

図であり、図1AはSSDのブロック図、図1Bは制御テーブルの構成図である。図1中、図12乃至図14と同じものは、同一符号で示してある。

【0026】本発明は上記の課題を解決するため、次のように構成した。

(1)、記憶媒体として半導体メモリ(TM8)を使用し、この半導体メモリ(TM8)を、所定の容量を持つ複数の論理ドライブDRV0、DRV1、・・・に分割して使用することにより、ディスク装置の入出力処理を仮想的に実行する半導体ディスク装置2の保守方法において、該半導体ディスク装置2の内部に、上記論理ドライブの制御情報格納用の制御テーブル9を、各論理ドライブDRV0、DRV1、・・・毎に設定すると共に、この制御テーブル9を、階層化したテーブルとして設定し、メモリの変更時に、各論理ドライブDRV0、DRV1、・・・毎に、上記制御テーブル9を更新処理することにより、該当する論理ドライブ以外の論理ドライブを、上位装置1からアクセス可能な状態で、メモリ変更処理を行うようにした。

【0027】(2)、上記構成(1)において、制御テーブル9を、制御情報の初期設定値を格納する基本制御テーブルと、該基本制御テーブルと同じ構成の増設用制御テーブルとで構成し、これら基本制御テーブルと、増設用制御テーブルに、次の制御テーブルのアドレスを設定することにより、階層化した。

【0028】(3)、上記構成(1)において、オンラインで稼働中の上記半導体ディスク装置において、メモリを増設する際、半導体ディスク装置2のプロセッサ(CA3、SVP6)では、外部からの指示により、上記制御テーブル9を参照して、増設する論理ドライブの制御情報を獲得し、増設する論理ドライブのみを、オフラインにして、増設部のフォーマット処理を行うと共に、増設した論理ドライブの制御テーブル9に、制御情報の値を再定義し、その後、増設部の論理ドライブをオンラインに戻すようにした。

【0029】(4)、上記構成(1)において、オンラインで稼働中の上記半導体ディスク装置において、障害の発生したメモリ修復(交換)処理を行う際、半導体ディスク装置2のプロセッサ(CA3、SVP6)では、外部からの指示により、上記制御テーブル9を参照して、修復する論理ドライブの制御情報を獲得し、修復する論理ドライブのみをオフラインにして、外部へ通知し、その後、障害メモリの交換終了の指示を受けたプロセッサ(CA3、SVP6)では、上記制御テーブル9を参照して、オフラインにした論理ドライブのフォーマット処理を行い、該フォーマット処理終了後、該当する論理ドライブを、オンラインに戻すようにした。

【0030】

【作用】上記構成に基づく本発明の作用を、図1に基づいて説明する。各論理ドライブに対し、基本制御テ

ブルを図1Bのように設定し、この基本制御テーブルと同じ構成のテーブルを、増設用制御テーブルとして設定する。増設用制御テーブルは、ドライブ、或いはシリンダを増設する時に使用する。

【0031】そして、制御テーブルに「次の制御テーブルのアドレス」の項を設け、この項に、次のテーブルの先頭アドレスを設定出来るようにして、該制御テーブルを階層化する。

【0032】また、基本制御テーブルには、「合計シリンダ数」(例えばN1)が設定出来るようになっており、この値によりそのドライブの有無を決めることが出来る。このようにして階層化した制御テーブル9により、論理ドライブに属するメモリ(TM8)の範囲を動的に現す事が可能となる。

【0033】上記のような階層化された制御テーブル9を設定した半導体ディスク装置2において、メモリを増設時等に、例えば、SVP6により、上記制御テーブル9が更新されると、SVP6からCA3へ割り込みを行う。

【0034】CA3ではこの割り込み処理において、制御テーブルが更新されたことを認識し、更新された制御テーブル9の制御情報を用いて、更新対象メモリ(TM8)部のフォーマット処理を行う。

【0035】この場合、更新対象の論理ドライブのみをオフラインとし、更新対象外の論理ドライブは、オンライン状態で、上記処理を行う。また、メモリ障害時の修復処理時においても、上記と同様に、修復する論理ドライブのみ、オフラインとし、その他の論理ドライブは、オンラインのまま、修復処理を行う事が出来る。

【0036】以上の通り、メモリの変更(増設等)の際に、装置をオンライン状態にしたまま、該当する論理ドライブのみをオフラインにして、処理を行うことが出来る。従って、装置を連続使用出来るから、装置の稼働効率(利用効率)が向上し、処理時間も短縮出来る。

【0037】

【実施例】以下、本発明の実施例を図面に基づいて説明する。図2～図11は、本発明の実施例を示した図であり、図2は半導体ディスク装置のブロック図、図3は基本制御テーブルとTMの関係例を示した図、図4は増設後の制御テーブルの構成例及びTMの関係例を示した図、図5は実施例1(ドライブを増やす場合)の制御テーブル例を示した図、図6は実施例1のTMの構成例を示した図、図7はSVPより増設指示を出して増設する場合の処理フローチャート、図8はSVPより指示を出して増設する場合の処理フローチャート、図9は実施例2(シリンダを増やす場合)の制御テーブル例を示した図、図10は実施例2の説明図(TMの構成例)、図11はメモリ障害時の修復(交換)を行う場合の処理フローチャートである。

【0038】図2～図11中、図1、及び図12乃至図

7

14と同じものは、同一符号で示してある。また、10は比較用制御テーブル（制御テーブル9と同じ構成）を示す。

【0039】①：SSD（半導体ディスク装置）の構成の説明・・・図2参照

図示のように、SSD2は、CA3、SVP6、CS4、CS5、TM8、PNL7等で構成され、ホストシステムであるCPU1に接続して使用される。

【0040】そして、上記CS4には制御テーブル9を設定し、CS5には制御テーブル9と、比較用制御テーブル10とを設定する。この比較用制御テーブル10は、制御テーブル9と同じ構成であり、外部（PNL7等）からの制御情報に基づいてSVP6が値をセットする。なお、CS5以外の構成は、上記従来例と同じなので、説明は省略する。

【0041】②：基本制御テーブルと、TM8の関係例の説明・・・図3参照

本実施例では、各論理ドライブに対し、各制御テーブル9に、図3に示したような基本制御テーブル（最初に設定した制御テーブル）を設定する。また、各制御テーブル9に、基本制御テーブルと同じ構成のテーブル（増設用テーブル）を、メモリの増設用として設定する。

【0042】図3に示した制御テーブル9の各項目は、次の通りである。すなわち、「Total CYLNUM」は、当該論理ドライブの合計のシリンダ数（増設する度に変更）（N1、N2、・・・）を示し、「Top Adrs-Basic」は当該論理ドライブの先頭アドレス（最初に設定した数値のまま）{（a）、（b）、・・・}を示す。

【0043】また、「Basic CYLNUM」は、当該制御テーブル9で設定するシリンダ数（増設テーブルでは、増設するシリンダ数）（N1、N2、・・・）を示し、「ADD Pointer」は当該論理ドライブの制御テーブル9の次の制御テーブルのアドレスを示す。なお、制御テーブル9が、基本制御テーブルのみの場合、及び当該ドライブの最終増設テーブルは、「0」とする。

【0044】上記のように、増設用テーブルは、シリンダ数を増やす時に使用し、制御テーブル9の「ADD Pointer」に、その制御テーブルの次の制御テーブルのアドレスを示し、「ADD Pointer」が「0」の時は、そのテーブルが最終制御テーブルとなる。そして、この「ADD Pointer」を使い、制御テーブル9を階層化する。

【0045】また、各ドライブの基本制御テーブルの「Total CYLNUM」によって、その論理ドライブの有無を決める（その値が「0」の時は、その論理ドライブは構成しない）。これにより、論理ドライブDRV0～DRV4に属するTM8の範囲を動的に現せる。

8

【0046】この制御テーブル9の改版が行われた時は、CA3に対する割り込み処理において、CA3が認識し、更新された制御テーブル9のフォーマットにより、更新対象のTM8を再フォーマット処理する。この場合、更新対象外の論理ドライブは、オンラインのまま、処理を続けることが出来る。

【0047】③：増設後の制御テーブルの構成及び、TMの関係例の説明・・・図4参照

上記図3に示した制御テーブルを増設した場合の制御テーブル及び、TM8の構成例を図4に示す。

【0048】この例では、論理ドライブDRV0に、M1の容量（シリンダ数）を増設し、論理ドライブDRV2を新しく増設した例である。この場合、増設前（図3の状態）は、DRV0のシリンダ数はN1、DRV1のシリンダ数はN2、DRV2のシリンダ数はN3=0、DRV3のシリンダ数はN4=0である。

【0049】そして、増設後は（図4の状態）、DRV0のシリンダ数はN1+M1（M1：増設分）、DRV1のシリンダ数はN2、DRV2のシリンダ数はN3（0でない数）、DRV3のシリンダ数はN4=0となる。

【0050】従って、DRV0の「Total CYLNUM」は、M1を増設するので、N1+M1の値を設定する。また、論理ドライブDRV0の基本制御テーブルの「ADD Pointer」は増設テーブルの先頭アドレスを示しており、論理ドライブDRV0の基本制御テーブル以外の「ADD Pointer」は、次のテーブルが無いために、「0」を設定する。

【0051】なお、当該ドライブが存在しない場合は、基本制御テーブルの「Total CYLNUM」に「0」を設定する。ところで、コンピュータシステムにおける保守のニーズとして、活性保守（装置の電源を落とさずに、部品交換を行い、システム運転を停止させないで、保守する）が要求されている中で、例えば、電源投入のまま、プリント板を挿抜する手法が開発されている。

【0052】本実施例では、上記の手法を利用して、SSD内のメモリプリント板（論理ドライブ）の増設削除、或いは、修復を行う。また、上記メモリの増設方法には、論理ドライブ数を増設する方法と、各論理ドライブのシリンダ数を増設する場合と、これら2つを組み合わせた方法との3つの方法がある。

【0053】以下、具体的な実施例について、詳細に説明する。

④：実施例1（ドライブ数を増設する方法）の説明・・・図5、図6、図7、図8参照

④-1：制御テーブル及び、TMに基づく説明・・・図5、図6参照

この例は、最大ドライブ数8のSSDにおいて、4KByteのメモリを、通常ドライブ数4（DRV0～DR

V3)、1シリンダ当たり、0x80Byte(128Byte)で、各ドライブに、DRV0-4シリンダ、DRV1-6シリンダ、DRV2-12シリンダ、DRV3-10シリンダ、の条件で使用していたものに、2KByteのメモリを増設し、DRV4-6シリンダ、DRV5-2シリンダ、DRV6-8シリンダ、の3つのドライブを増設する例である。

【0054】図5に示した制御テーブル例において、ドライブDRV0、DRV1、DRV2、DRV3が今まで使用していた論理ドライブ(増設前のドライブ)であり、ドライブDRV4、DRV5、DRV6が増設する論理ドライブである。

【0055】ドライブDRV0~DRV3の制御テーブルにおいて、「Top Adrs Basic」或いは、「Basic CYLNUM」に変更が有った場合は、論理ドライブの設定のやり直しと判断し、SVP6において、そのことを判断させ、従来と同じ処理で論理ドライブを設定する。

【0056】上記処理時のTM8の構成を図6に示す。図示のように、増設後のTM8には、DRV4、DRV5、DRV6が増設されている。なお、この判断は、CA3の割り込み処理においても実行させる事が出来るが、SVP6において、処理したほうが実行処理が少なくなるので、この例では、SVP6で判断させる。

【0057】また、DRV0~DRV3の制御テーブルにおいて、「Top Adrs Basic」或いは、「Basic CYLNUM」以外の所に変更がある場合は、後述する。

【0058】④-2:フローチャートに基づく説明・・・図7、図8参照

図7、図8は、SVPより指示を出して、増設する場合の処理フローチャートであり、以下、図7、図8に基づいて説明する。なお、図のS1~S14は、各処理番号を示す。

【0059】(S1):まず、SVP6は、外部からの制御情報に基づき、CS5内の比較用制御テーブル10に値をセットする。

(S2):SVP6より増設を指示する。

【0060】(S3):SVP6により、今設定されている(オンライン中の)制御テーブル9と、比較用制御テーブル10とを比べ、増設部以外に変更点は無いかを判断する。もし、変更点があった場合は、従来例(上記従来例の②-4からの処理)と同じように処理する。

【0061】(S4):しかし、変更点があった場合には、SVP6により、比較用制御テーブル10に、正常な値がセットされているか(アドレスの重複が無いかなど)を確認する。もし、重複があれば、制御テーブルの再定義を促す。

【0062】(S5):比較用制御テーブル10に、正常な値がセットされている場合には、SVP6より、C

A3に対して、割り込みを行う。

(S6):この時、SVP6では、CS5内の制御テーブル9に値(制御情報の値)を再定義し、通常運用を行う。

【0063】(S7):CA3は、上記割り込みにより、各ドライブに対して、以下の処理を実行する。

(S8):CA3では、基本ドライブ分(最大ドライブ数)処理を行ったかどうかを調べ、処理が完了していれば、通常運用を行う。

【0064】(S9):しかし、処理が完了していなければ、CA3は、今設定されている(オンライン中の)制御テーブル9と、これから設定する制御テーブルとを比べ、どのように(ドライブを幾つ、シリンダが幾つ)フォーマットするか調べる。

【0065】すなわち、制御テーブルに変更があるかを調べ、無ければ、次のドライブの処理に移る。

(S10):CA3は、制御テーブル9内の「Total CYLNUM」が「0」かを調べる。

【0066】(S11):もし、「Total CYLNUM」が「0」ならば、当該ドライブを削除(使用しない)するので、当該ドライブを、オフラインにする。

(S12):そして、CS4内の制御テーブル9に値を再定義する。

【0067】(S13):また、「Total CYLNUM」が「0」でない場合は、増設部のみオフラインにして、該増設部にのみ、フォーマット処理を行う。

(S14):終了したら、増設したドライブに対してCA3がエミュレートを行えるように、CS4内の制御テーブル9に値を再定義する。

【0068】(S15):CA3は増設したドライブをオンラインにする。

以上の各処理により、増設部のみフォーマットして、他はオンラインのまま、メモリの増設を行うことが出来る。

【0069】なお、この例では、3つのドライブを増設したが、制御テーブルの基本テーブルの数を変えることにより、増設出来るドライブの数を設定する。また、増設出来るドライブ数を越えなければ、何度でも増設が可能である。

【0070】⑤:実施例2(各ドライブのシリンダ数を増設する例)の説明・・・図9、図10参照

図9はシリンダを増設する場合の制御テーブルの例である。また、図10は、実施例2におけるTM8の構成例であり、図10Aは増設前のTM8の構成例、図10Bは増設後のTM8の構成例、図10CはCA3から見たTM8の考え方(エミュレートの仕方)を示す。

【0071】⑤-1:制御テーブル及び、TMによる説明・・・図9、図10参照

この例は、最大ドライブ数5のSSDにおいて、4KByteのメモリを、通常ドライブ数4(DRV0~DR

V3)、1シリンダ当たり、0x80Byte(128 Byte)で、各ドライブに、DRV0-4シリンダ、DRV1-6シリンダ、DRV2-12シリンダ、DRV3-10シリンダ、の条件で使用していたものに、2KByteのメモリを増設し、DRV0、DRV1、DRV3の3つのドライブに、それぞれ、6、2、8シリンダづつ増設する例である。

【0072】図示のように、DRV0~DRV3までが、今まで使用していたドライブであり、そのうち、DRV0、DRV1、DRV3にそれぞれシリンダを増設する。DRV0~DRV3の制御テーブルにおいて、

「Top Ads Basic」或いは「Basic CYLNUM」に変更が有った場合は、論理ドライブの設定のやり直しと判断し、SVP6において、そのことを判断させ、従来と同じ処理において、論理ドライブを設定する(理由は上記実施例1と同じ)。

【0073】⑤-2:フローチャートに基づく説明・・・図7、図8参照
上記図7、図8は、実施例2でも共通しているもので、以下、図7、図8に基づいて、実施例2の処理を説明する。なお、図のS1~S14は、各処理番号を示す。

【0074】(S1):まず、SVP6は、外部からの制御情報に基づいて、CS5内の比較用制御テーブル10に値をセットする。

(S2):SVP6より増設を指示する。

【0075】(S3):SVP6により、今設定されている(オンライン中の)制御テーブル9と、比較用制御テーブル10とを比べ、増設部以外に変更点はないかを判断する。もし、変更点があった場合は、従来例と同じように処理する。

【0076】(S4):しかし、変更点が無ければ、SVP6により、比較用制御テーブル10に、正常な値がセットされているか(アドレスの重複が無い等)を確認する。もし、重複があれば、制御テーブルの再定義を促す。

【0077】(S5):比較用制御テーブル10に、正常な値がセットされている場合には、SVP6より、CA3に対して、割り込みを行う。

(S6):この時、SVP6では、CS5内の制御テーブル9に値を再定義し、通常運用を行う。

【0078】(S7):CA3は、上記割り込みにより、各ドライブに対して、以下の処理を実行する。

(S8):CA3では、基本ドライブ分(最大ドライブ数)処理を行ったかどうかを調べ、処理が完了していれば、通常運用を行う。

【0079】(S9):しかし、処理が完了していなければ、今設定されている(オンライン中の)制御テーブル9と、これから設定する制御テーブルとを比べ、どのように(幾つシリンダを増設するか)フォーマットするか調べる。

【0080】この時、制御テーブル9の「ADD Pointer」を調べ、「0」でない時は、階層的に次のテーブルを調べていき、「0」になるまで調べる。そして、制御テーブルに変更があるかを調べ、なければ、次のドライブの処理に移る。

【0081】(S10):CA3は、制御テーブル9内の「Total CYLNUM」が「0」かを調べる。

(S11):もし、「Total CYLNUM」が「0」ならば、当該ドライブを削除(使用しない)するので、当該ドライブを、オフラインにする。

【0082】(S12):そして、CS4内の制御テーブル9に値を再定義する。

(S13):また、「Total CYLNUM」が「0」でない場合は、増設部のみオフラインにして、該増設部にのみ、フォーマット処理を行う。

【0083】(S14):終了したら、増設したドライブに対してCA3がエミュレートを行えるように、CS4内の制御テーブル9に値を再定義する。

(S15):増設したドライブをオンラインにする。

【0084】以上の各処理により、増設部のみフォーマットして、他はオンラインのまま、メモリの増設を行うことが出来る。なお、この例では、増設は1回だったが、「ADD Pointer」を次々に指定することにより、何段階にも、階層的に制御テーブルを繋げることが出来、それにより、何度でも、増設が可能である。

【0085】なお、上記(S14)の処理において、実際は、図10A、図10BのようにTM8は構成されるが、CS4内の制御テーブル9を用いてエミュレートする際、CA3は、図10CのようにTM8を見る。

【0086】例えば、DRV0を例にすれば、DRV0の0x220Byte目より、アクセスしようとする場合、CA3によって、0x1020番地よりアクセスするようにエミュレートする。

【0087】

⑥:実施例3(上記2つの方法を合わせた方法)

この例は、上記2つの方法、すなわち、ドライブの増設と、シリンダの増設とを合わせて行う例である。

【0088】実施例3は、上記実施例2の制御テーブルを複数個余分に持つことによって、実現出来る。すなわち、上記実施例1と実施例2を合わせた処理なので説明は省略する。

【0089】⑦:実施例4(メモリ障害時にメモリを修復する場合の実施例)・・・図11参照

この例は、TM8の或るアドレスにおいて、メモリ障害が発生した時の修復処理の例である。以下、図11の処理フローチャートに基づいて、実施例4の処理を説明する。なお、図11のS21~S31は、各処理番号を示す。

【0090】(S21):PNL7より障害メモリのアドレスを指定して、修復処理を行うことをSVP6へ指

13

示する(例えば、カスタマエンジニアが指示する)。

(S22): SVP6はCS5内の制御テーブル9を調べ、障害メモリのアドレスが、どのドライブに割り振られているかを調べる。

【0091】(S23): SVP6は、CA3に対して、上記(S22)で調べたドライブをオフラインにする割り込みを行う。

(S24): CA3の割り込み処理において、指示されたドライブをオフラインにする。

【0092】(S25): CA3は、オフラインになったことを、SVP6に通知する。

(S26): SVP6は、オフラインになったことを確認したら、PNL7を通して、PNL7を操作している人(例えば、カスタマエンジニア)へ知らせる。

【0093】(S27): 例えば、カスタマエンジニアは、障害メモリを新品と交換する。

(S28): 例えば、カスタマエンジニアは、PNL7より、オフラインにしたドライブを、オンラインにするように、SVP6へ指示する(交換終了指示)。

【0094】(S29): SVP6は、ドライブをオンラインにするように、CA3へ割り込みを行う。

(S30): CA3では、割り込み処理において、オフラインにしたドライブの制御テーブル9(CS4内の制御テーブル)を、参照して、ドライブのフォーマット処理を行う。

【0095】(S31): その後、CA3は、フォーマット処理したドライブを、オンラインにする。

以上の各処理により、障害メモリ部のドライブのみオフラインにするだけで、障害メモリの修復を行う事が可能となる。

【0096】

【発明の効果】以上説明したように、本発明によれば次のような効果がある。

(1)、装置をオフラインにすることなく、メモリを増設したり、或いは障害メモリの修復をしたりする事が出来る。従って、半導体ディスク装置の連続使用が出来る、該装置の稼働効率(利用効率)が向上する。

14

【0097】(2)、装置をオフラインにすることなく、メモリを増設したり、或いは障害メモリの修復をしたりする事が出来る。従って、メモリを増設時、或いは障害メモリの修復時の処理時間が短縮できる。

【図面の簡単な説明】

【図1】本発明の原理説明図である。

【図2】本発明の実施例における半導体ディスク装置のブロック図である。

【図3】本発明の実施例における基本制御テーブルとTMの関係例である。

【図4】本発明の実施例における増設後の制御テーブルの構成例及びTMの関係例である。

【図5】実施例1(ドライブを増やす場合)の制御テーブル例である。

【図6】実施例1のTMの構成例である。

【図7】SVPより増設指示を出して増設する場合の処理フローチャート(その1:SVPの処理)である。

【図8】SVPより増設指示を出して増設する場合の処理フローチャート(その2:CAの処理)である。

【図9】実施例2(シリンダを増やす場合)の制御テーブル例である。

【図10】実施例2の説明図(TMの構成例)である。

【図11】メモリ障害時のメモリの修復(交換)を行う場合の処理フローチャートである。

【図12】従来例の説明図(その1)である。

【図13】従来例の説明図(その2)である。

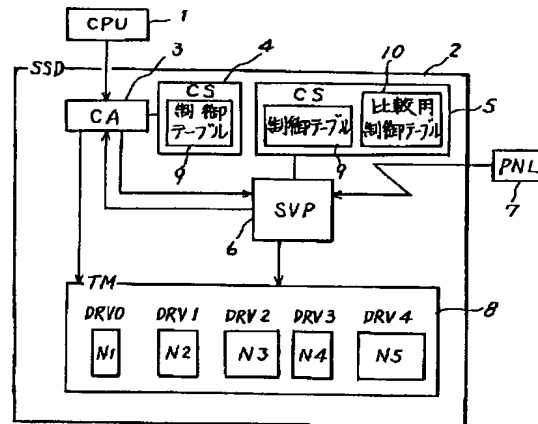
【図14】従来例の説明図(その3)である。

【符号の説明】

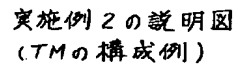
- 1 CPU(中央処理装置)
- 2 SSD(半導体ディスク装置)
- 3 CA(チャネルアダプタ)
- 4、5 CS(コントロールストレージ)
- 6 SVP(サービスプロセッサ)
- 7 PNL(メンテナンスパネル)
- 8 TM(トラックメモリ)
- 9 制御テーブル

【図 2】

実施例における半導体ディスク装置のブロック図

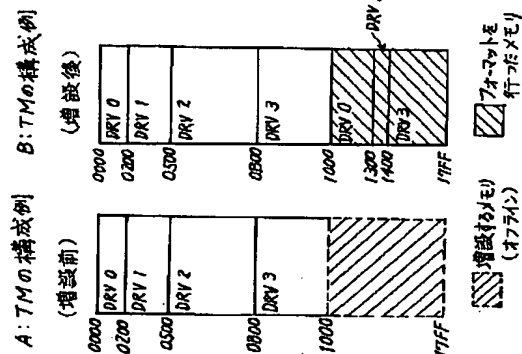


【例 10】



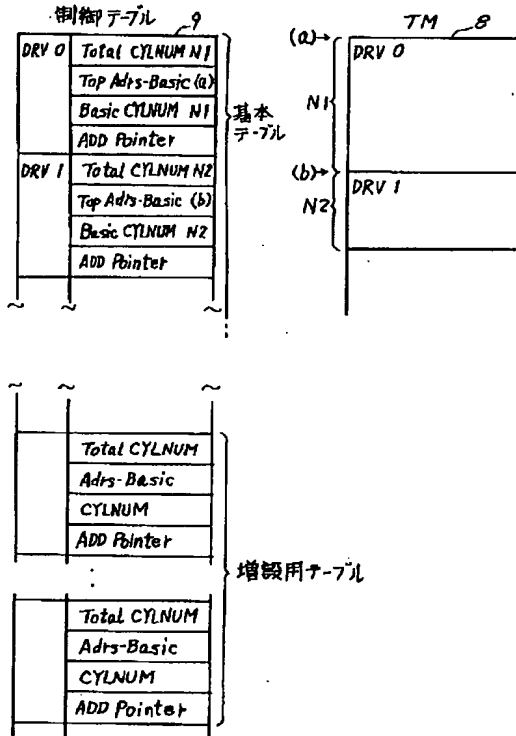
C: CAから見たTM
の考え方

0000	DRV 0
1000	
0200	DRV 1
1300	
0500	DRV 2
0800	
0900	DRV 3
1400	
1500	

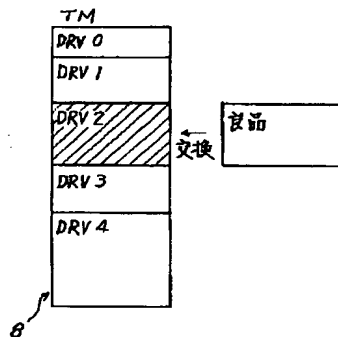


【図3】

基本制御テーブルとTMの関係例



【図14】

従来例の説明図 (その3)
(メモリ障害の修復時)

【図5】

実施例1: (ドライブを増やす場合)の制御テーブル例

	増設前	増設後
容量(Byte/シリンダ)		
全体	4K kByte	6 kByte
DRV 0	512 / 4	512 / 4
DRV 1	768 / 6	768 / 6
DRV 2	1536 / 12	1536 / 12
DRV 3	1280 / 10	1280 / 10
DRV 4	0 / 0	768 / 6
DRV 5	0 / 0	236 / 2
DRV 6	0 / 0	1024 / 8
DRV 7	0 / 0	0 / 0

アドレス 0x0000~0x0FFF 0x0000~0x7FFF
1シリンダ 0x80 Byte (128 Byte)
ドライブ数 4(0~3) 7(0~6)

DRV	Total CYLNUM
	Top Adrs-Basic
	Basic CYLNUM
	ADD Pointer

増設前

制御テーブル


増設後

DRV 0	4
	0000
	4
	0
DRV 1	6
	0200
	6
	0
DRV 2	12
	0500
	12
	0
DRV 3	10
	0800
	10
	0

DRV 4	0
	0000
	0
	0
DRV 5	0
	0000
	0
	0
DRV 6	0
	0000
	0
	0
DRV 7	0
	0000
	0
	0

DRV 0	4
	0000
	4
	0
DRV 1	6
	0200
	6
	0
DRV 2	12
	0500
	12
	0
DRV 3	10
	0800
	10
	0

DRV 4	6
	1000
	6
	0
DRV 5	2
	1800
	2
	0
DRV 6	8
	1400
	8
	0
DRV 7	0
	0000
	0
	0

 交換を行ったテーブル

【図4】

増設後の制御テーブルの構成例及びTMの関係例

最大ドライブ数 4ドライブ

増設前 (図3参照)

DRV 0 シリンダ数 ... N1

1 ... N2

2 ... N3=0

3 ... N4=0

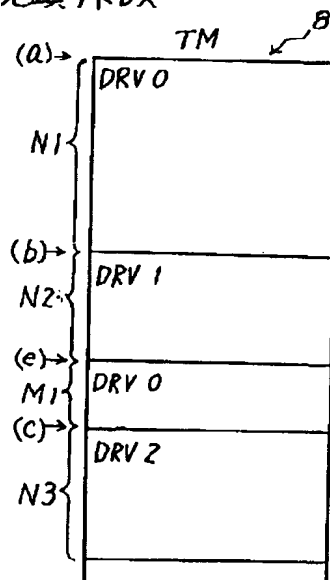
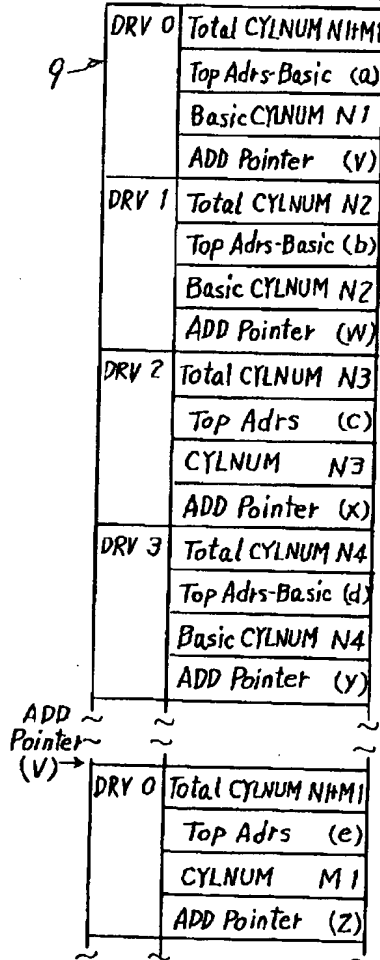
N1~N4, M1 ... それぞれのシリンダ数

(a) ~ (d) ... それぞれのドライブの先頭アドレス

(V) ~ (Z) ... それぞれの増設テーブルの先頭アドレス

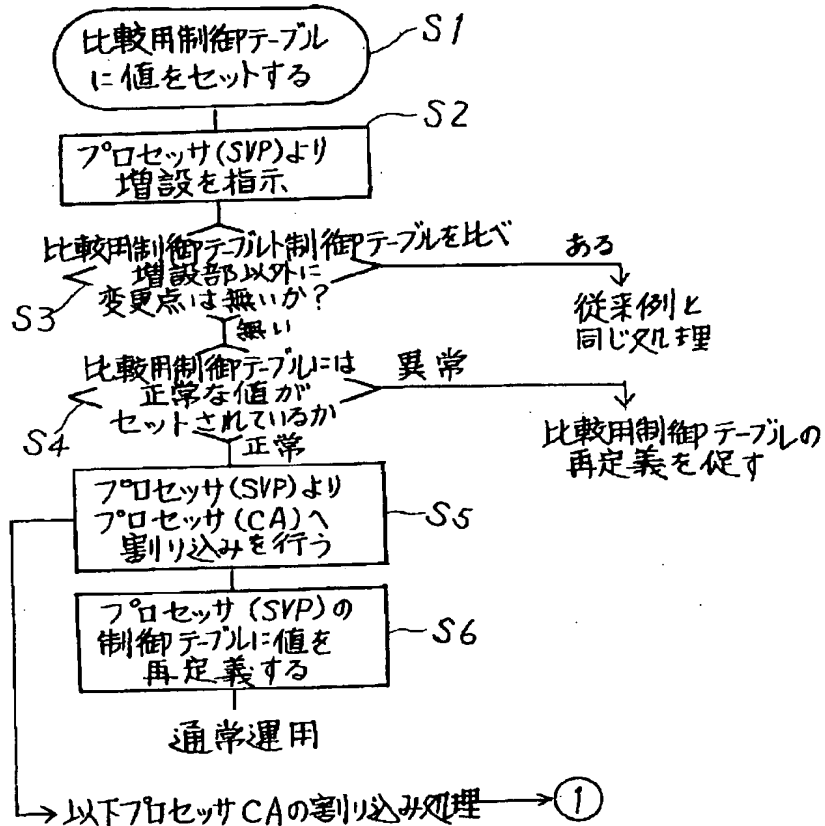
(この例では(W)~(Z)の値は0とする。(V)のみ値をセットする)

(e) ... DRV 0 の増設メモリの先頭アドレス



【図7】

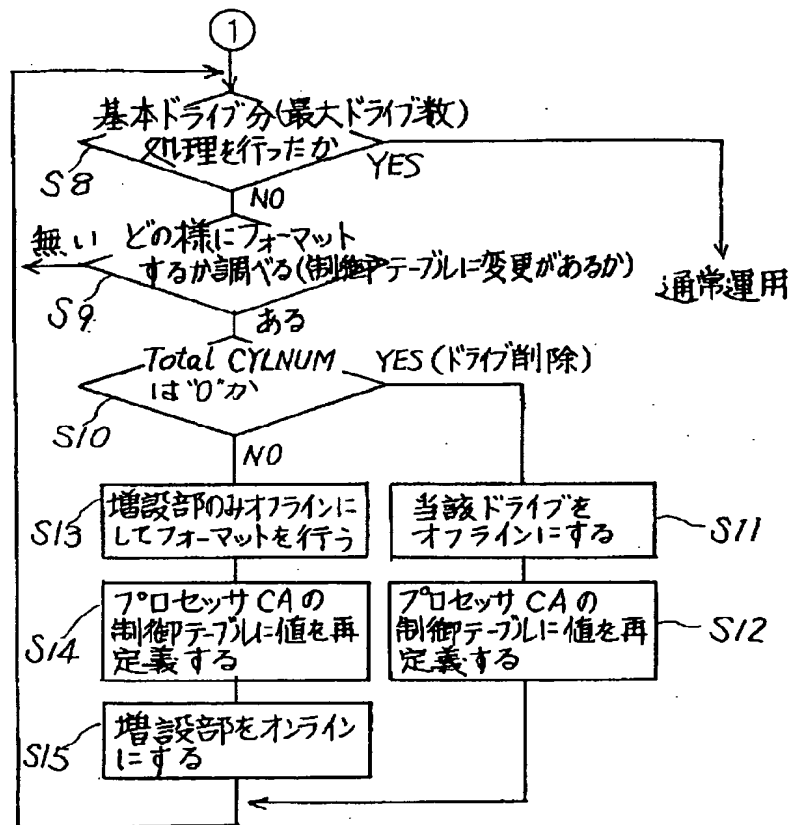
SVPより増設指示を出して増設する場合の
処理フローチャート（その1: SVPの処理）



(S1~S6: SVPの処理)

【図8】

SVPより増設指示を出して増設する場合の
処理フローチャート（その2:CAの処理）

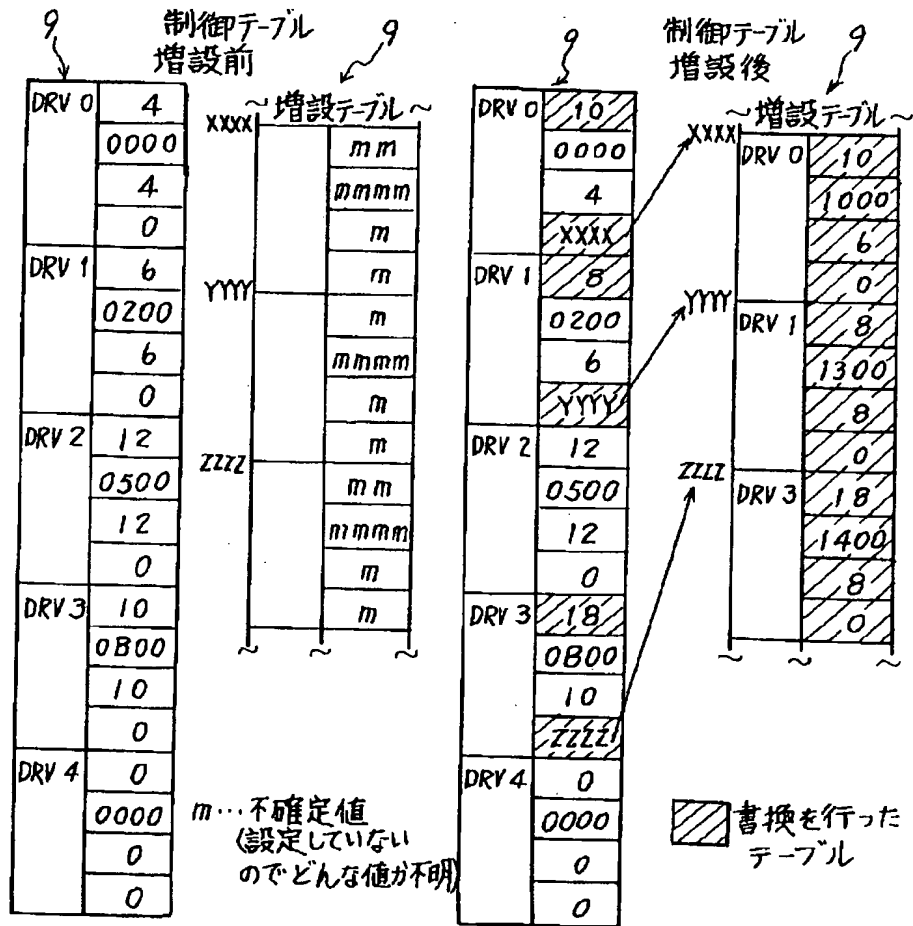


(S8～S15: CAの処理)

【図9】

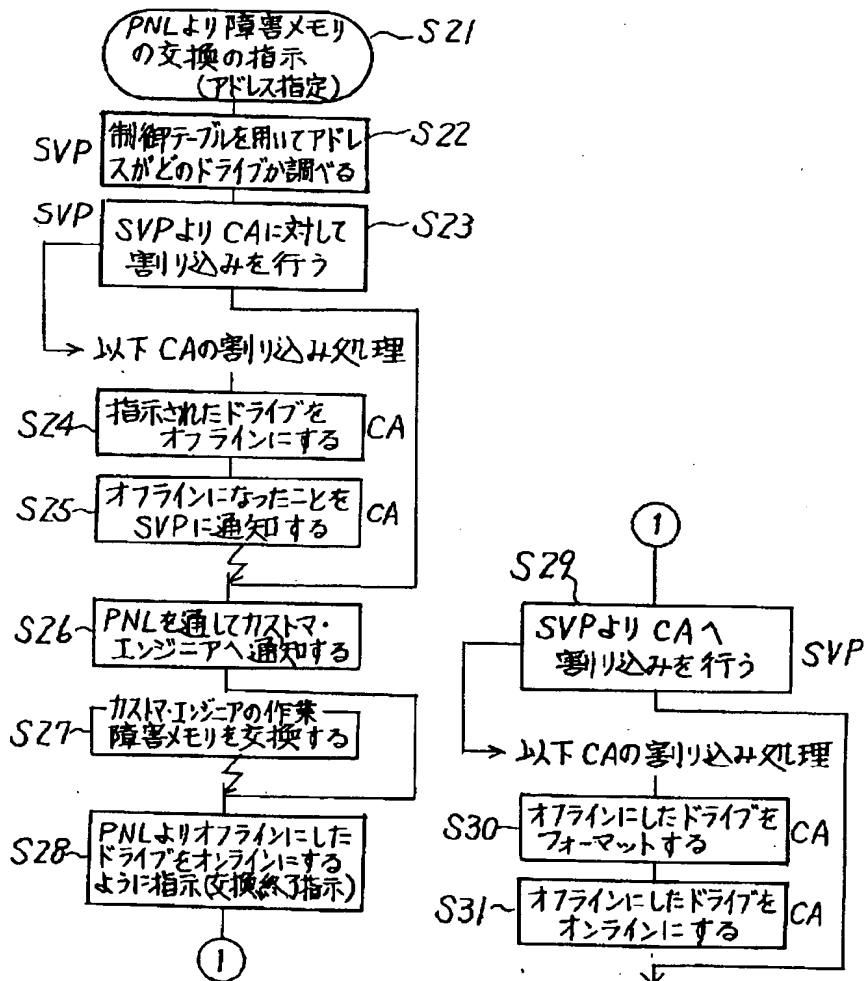
実施例2: (シリンダを増やす場合)の制御テーブル例

	増設前	増設後	基本制御テーブル
容量(Byte/シリンダ)			
全体	4 KByte	6 KByte	
DRV0	512 / 4	1280 / 10	
DRV1	768 / 6	1024 / 8	
DRV2	1536 / 12	1536 / 12	Total CYLNUM
DRV3	1280 / 10	2304 / 18	Top Adrs-Basic
DRV4	0 / 0	0 / 0	Basic CYLNUM
アドレス	0x0000~0x0FFF	0x0000~0x17FF	ADD Pointer
1シリンダ	0x80 Byte (28 Byte)		
ドライブ数	4 (0~3)	4 (0~3)	



【図11】

メモリ障害時のメモリの修復(交換)を行う場合の
処理フローチャート

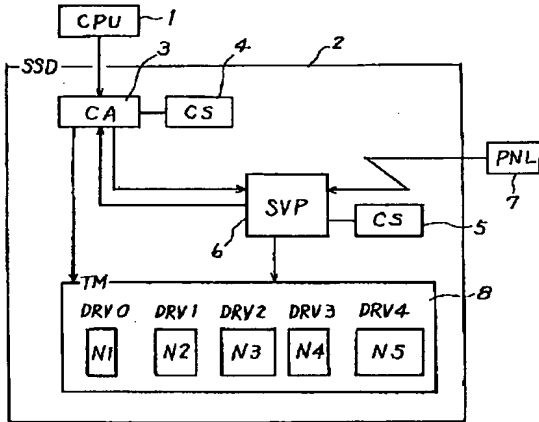


S21, S26~S28: カスタムエンジニア
の処理

【図12】

従来例の説明図（その1）

A: 半導体ディスク装置のブロック図



B: 制御テーブル例

	CYLNUM
DRV 0	N1
DRV 1	N2
DRV 2	N3
DRV 3	N4
DRV 4	N5

【図13】

従来例の説明図（その2）

（シリンダ数を増やす場合）

